

Protokoll zum 10.11.2006

Oliver Gries

Atomare / Komplexe Konzepte:

Der Unterschied zwischen atomaren und komplexen Konzepten ist rein syntaktisch: Atomare Konzepte sind syntaktisch nicht weiter zerlegbar, es sind einzelne Symbole. Komplexe Konzepte sind syntaktisch strukturiert und aus mehreren Symbolen aufgebaut.

base symbols / name symbols:

Base und name symbols sind sowohl logische Symbole als auch atomare Konzepte. Base symbols stehen nur auf der rechten Seite einer TBox. Name symbols dürfen nur einmal auf der linken Seite und beliebig oft auf der rechten Seite einer TBox-Definition stehen. Sie sind relativ zur TBox zu interpretieren.

Teil I) Vorbemerkungen zu Beschreibungslogiken

12) Was ist die Unique-Name-Assumption? Welche Gründe sprechen wohl dafür, sie im Kontext von beschreibungslogischen ABoxen zu machen, und welche Gründe sprechen dagegen, sie generell in der Prädikatenlogik zu machen ?

Wenn die Unique-Name-Assumption(UNA) gilt, dann repräsentieren unterschiedliche Bezeichner (Strings, Bezeichner für Konstanten) unterschiedliche Objekte. Übertragen auf die Datenbankwelt bedeutet die UNA, dass verschiedene Schlüssel verschiedene Objekte bezeichnen: Jeder Schlüssel denotiert eine andere Person, Vertrag, etc., so dass kein Objekt doppelt repräsentiert ist. Bei den Beschreibungslogiken ist es üblich, und es wird erwartet, die UNA in der ABox zu benutzen. Wäre das nicht der Fall, wäre es sehr aufwendig, alle ungleichen Einträge in der ABox als verschieden festzulegen. In der TBox wird die UNA nicht vorausgesetzt. Es wäre eine Inferenzaufgabe, zu testen, ob zwei unterschiedliche Konzepte die gleiche Menge von Objekten repräsentieren.

13) Gib Beispiele von Konzepten der Studien-Domäne an, die sich unter Verwendung der Konstruktoren 'one-of' oder 'fills' spezifizieren lassen.

Es gibt Ansätze mit strikter Trennung von Konzeptebene und dem Weltmodell. Möchte man mit der T-Box auf die A-Box zugreifen, so ist zunächst der 'one-of'-Operator geeignet, wenn man alle Elemente der Menge kennt und weiß, es gibt nicht mehr. Nötig und nützlich ist der Operator aber nicht immer. In der

Studiendomäne wäre das Konzept 'Note' geeignet. In der abstrakten Syntax könnte man hier den 'one-of'-Operator folgendermaßen einbauen:

Note $\equiv \text{oneOf}(\{'1', \dots, '5'\})$
 Prüfungsversuch $\sqsubseteq (\geq 1) \text{ hatNote} \sqcap (\leq 1) \text{ hatNote} \sqcap \forall \text{ hatNote. Note}$
 bestandenerPrüfungsversuch $\equiv \text{Prüfungsversuch} \sqcap \forall \text{ hatNote. bestandeneNote}$
 bestandeneNote $\equiv \text{oneOf}(\{'1', \dots, '4'\})$

Ein Beispiel für den 'fills'-Operator wäre:

nicht bestandenerPrüfungsversuch $\equiv \text{Prüfungsversuch} \sqcap \forall \text{ hatNote:5}$

14) Was bedeutet die Möglichkeit der Reduktion 'klassischer' Inferenzaufgaben auf Subsumptions- oder Erfüllbarkeitsaufgaben für die geplante Implementation des Reasoning-Servers ?

Die Reduktion ist für die Implementation sehr praktisch. Es muss keine Kopie der Inferenzprogramme angefertigt und anschließend geändert werden. Jeder Fehler wird so nur einmal gemacht.

15) Was ist die Expansion eines Konzeptes (oder allgemeiner einer Beschreibung) bezüglich einer TBox? Wie muss eine TBox ggf. umgeformt werden, damit die Expansion definiert (und eindeutig) ist? Welche Eigenschaften hat die Expansion einer Beschreibung, die es sinnvoll macht, sie anstelle der Original-Beschreibung zu verarbeiten ?

Die Expansion einer T-Box und eines Konzeptes ist eigentlich das Gleiche und wurde letztes Mal ausführlich behandelt.

16) Welche Möglichkeiten siehst Du, (eindeutige) Expansionen für das Konzept A bzgl. der TBoxen T1-T5 zu bilden:

T1 = $\{A \sqsubseteq D \sqcup C\}$

T2 lässt sich nicht expandieren, ist nicht deterministisch

T3 muss umgeformt werden, um expandiert werden zu können:

$B \sqsubseteq C \sqcap B^* \rightarrow T3 = \{A \sqsubseteq D \sqcup (C \sqcap B^*)\} = \{A \sqsubseteq (D \sqcup (C \sqcap B^*)) \sqcap A^*\}$

T4 erst zusammenfassen, dann normieren:

$= \{A \sqsubseteq (D \sqcup B) \sqcap C\} = \{A \sqsubseteq (D \sqcup B) \sqcap C \sqcap A^*\}$

T5 lässt sich nicht expandieren, weil auf der linken Seite von Konzeptdefinitionen atomare Konzepte stehen müssen

Exkurs: Gilt die Aussage zu T5 auch für Inklusionen ?

Sowohl $A \sqcap C \sqsubseteq D \sqcup B$ als auch $A \sqcup C \sqsubseteq D \sqcup B$ lassen sich umformen, so dass auf der linken Seite nur atomare Konzepte stehen. Das erste der beiden Axiome wurde ausführlich behandelt. Es gilt:

$$A \sqcap C \sqsubseteq D \sqcup B \Leftrightarrow A \sqsubseteq D \sqcup B \sqcup \neg C$$

Der Beweis wurde mit Venn-Diagrammen erstellt.

17) Verschiedene real existierende Reasoning-Server für Beschreibungslogiken (FaCT, Racer) implementieren die Prüfung von ABox-Konsistenz direkt und führen andere Aufgaben darauf zurück. Kannst Du dafür argumentieren, warum das sinnvoll ist?

Die Entwickler von Racer behaupten, ihr System sei effizient bei vollen ABoxen. FaCT-Algorithmen führen Tableau-Beweise mit minimalen ABoxen durch. Es soll geprüft werden, ob $\{C(a)\}$ konsistent ist.

18) Warum behandeln Baader und Nutt Rollenzuordnungen ('role assertions') in 2.2.4.3. so gut wie gar nicht? Unter welchen Umständen werden auch wir Rollenzuordnungen weitgehend ignorieren können?

Rollen werden nur dazu benutzt, um Konzepte zu definieren. Rollenkonstruktoren für die Erstellung von neuen Rollen werden bei uns aber nicht häufig angewendet. Wir hatten die inverse Rolle

$$\text{hatTeilnehmer}^- \equiv \text{besucht}$$

in der Studentendomäne benutzt. Bei dem Tableau-Verfahren gibt es bezüglich der Bildung von Rollen durch Rollenkonstruktoren Probleme: Bei $\forall R.C$ wissen wir nicht, ob es das jeweilige $R(x,y)$ gibt (siehe Aufgabe 24). Bei $\exists R.C$ und $\geq .R$ gibt es dieses Problem nicht, weil neue Konstanten erschaffen werden.

19) Wie verhalten sich Reasoning-Services, die Retrieval- und Realisierungsaufgaben bearbeiten, zu den anderen besprochenen Reasoning-Services? Wie kann eine einfache Lösung implementiert werden?

retrieval = alle Instanzen werden geprüft, ob sie Instanzen von dem jeweils gesuchten Konzept sind

realization = zu einem Objekt werden alle Konzepte genannt, die es erfüllt

Bei Datenbanken gibt es diese Reasoning-Services auch. Es ist aber gut, dass sie keine komplexen TBoxen benutzen. Bei ihnen ist wichtig, dass sie komplexe Anfragen schnell beantworten können

20) Welche Gemeinsamkeiten und Unterschiede bestehen zwischen Datenbanken auf der einen Seite und Logik-basierten WR-Ansätzen auf der anderen Seite hinsichtlich der Interpretation eines Bestandes an Aussagen und den damit verbundenen Möglichkeiten des Schließens ('reasoning').

Siehe Aufgaben 12) und 19).

21) ...

Teil II) Tableau-Algorithmen in Beschreibungslogiken

22) + 23) Was ist unter Struktur-basierter Verarbeitung zu verstehen? Welche Rolle kann der Inhalt einer TBox bei dem geschilderten Verfahren spielen? Was sind die Vor- bzw. Nachteile von Tableau-basierten Algorithmen gegenüber Struktur-basierten Algorithmen für die Verarbeitung von Beschreibungslogiken?

Ein strukturelles Verfahren bringt jedes zu untersuchende Konzept zunächst in eine Normalform und vergleicht dann die einzelnen Konjunkte. Gibt es z.B.

$$B1 \equiv A1 \sqcap \dots \sqcap Ak \sqcap \forall R1.C1 \sqcap \dots \sqcap \forall Rn.Cn \text{ und}$$

$$B2 \equiv A'1 \sqcap \dots \sqcap A'l \sqcap \forall R'1.C'1 \sqcap \dots \sqcap \forall R'm.C'm$$

in Normalform, versucht man festzustellen, ob $B1 \sqsubseteq B2$ gilt. Dazu wird nach und nach geprüft, ob alles, was in $B1$ verlangt wird, auch in $B2$ vorkommt. Der Nachteil ist, dass keine Existenzrestriktion ($\exists R.C$), kein \perp und keine Negation ($\neg A$) vorkommen kann. Die Ausdrucksmächtigkeit ist also stark begrenzt. Der Vorteil gegenüber dem allgemeineren Tableau besteht darin, dass Struktur-basierte Verfahren leichter zu behandeln sind: Hat man alle Konzepte erst einmal in eine Normalform gebracht, geht der Rest ganz schnell.

24) In 2.3.2. beschreiben Baader & Nutt die Konstruktion von Modellen zum Testen der (Un-)Erfüllbarkeit von Beschreibungen. Was unterscheidet diese Darstellung von Tableau-Verfahren, so wie Fitting sie beispielsweise darstellt?

Die Expansionsregeln der Tableau-Verfahren unterscheiden sich natürlich.

\sqcap : Wenn ein Zweig (ABox) $(C1 \sqcap C2)(x)$ enthält, aber nicht $C1(x)$ und $C2(x)$, werden letztere dem Zweig hinzugefügt (entsteht eine um $C1(x)$, $C2(x)$ erweiterte ABox).

\sqcup : Wenn ein Zweig $(C1 \sqcup C2)(x)$ enthält, aber weder $C1(x)$ noch $C2(x)$, werden

diese beiden in verschiedene Zweige hinzugefügt. Bei dieser Regel entstehen zwei neue ABoxen.

\exists : Wenn $(\exists R.C)(x) \in A$ und es ist nicht der Fall, dass sowohl $C(z)$ als auch $R(x,z) \in A \rightarrow A' = A \cup \{R(x,y), C(y)\}$ und y kommt nicht in A vor. Das Hinzufügen einer neuen Rollenassertion kann die folgende Expansion entscheidend beeinflussen.

\forall : $(\forall R.C)(x)$ und $R(x,y) \in A \rightarrow A' = A \cup \{C(y)\}$. Zu beachten ist, dass die Regel für alle möglichen Konstanten y , die zu x in der Relation R stehen, $C(y)$ hinzufügt. Diese Regel wird meistens dann angewendet, wenn zuvor mit der \exists -Regel (oder der \geq -Regel) $R(x,y)$ in den Zweig geschrieben wurde.

Im Gegensatz zu anderen Konzeptbildungsoperatoren mit Rollen können hier Probleme entstehen: Möchte man z.B. R aus zwei anderen Rollen S und T mit der Rollenkomposition $R \equiv S \circ T$ erzeugen, weiß der Beweiser während der Ausführung der \forall -Regel nicht, ob es $R(x,y)$ gibt, auch wenn $S(x,z)$ und $T(z,y) \in A$. Deswegen lassen sich hier Rollenkonstruktoren nicht immer erfolgreich anwenden.

\geq : $(\geq n.R)(x) \in A$, $R(x,z_i) (1 \leq i \leq n) \notin A$ und $z_i \neq z_j (1 \leq i < j \leq n) \rightarrow A' = A \cup \{R(x,y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$ und y_1, \dots, y_n kommen in A nicht vor. Die Regel wird nur dann ausgeführt, wenn nicht schon n Rollenassertionen $R(x,z_i)$ existieren, von deren Konstanten z_i man weiß, dass sie paarweise disjunkt sind. Bei der Ausführung der Regel werden sowohl n neue Relationen als auch die Ungleichheit der n neuen Konstanten in die ABox geschrieben.

Die Regel fügt die n Relationen vollständig hinzu, d.h. auch dann, wenn x vor der Ausführung der Regel bereits zu mehreren y_i in der Relation stehen sollte. Wenn z.B. x zu drei Konstanten in der Relation `hatTochter` steht, dann werden bei der Anwendung der \geq -Regel fünf neue Töchter hinzugefügt und x steht zu acht Konstanten in der Relation `hatTochter`.

\leq : Wenn $(\leq n.R)(x) \in A$, es $n+1$ Relationen $R(x,i)$ gibt und man nicht von allen Konstanten in diesen Relationen weiß, dass sie verschieden sind, dann erstelle einen neuen Zweig, in dem zwei dieser Konstanten gleichgesetzt werden. Diese Regel ersetzt immer nur eine Konstante. Bei $(\leq 5.R)$ werden 6er Gruppen gecheckt (alle Paare durchprobieren). Wenn ein Paar gleich ist wird nur eins ersetzt \rightarrow Optimierungspotenzial!

25) Die UNA bei Tableau mit ABoxen wäre eine Einschränkung. Die \leq -Regel wäre nicht mehr ausführbar, weil bereits alle Individuen disjunkt sind. Deswegen muss

explizit mit $x \neq y$ angegeben werden, dass zwei Individuen sich unterscheiden. Bei der Ausführung der \geq -Regel wird das beispielsweise gemacht.

26) Die Tableau-Expansionsregeln sind bei Baader & Nutt als Bedingungs-Aktionsregeln formuliert. Wie unterscheiden sich die vorliegenden Formulierungen von den Tableau-Regeln, wie sie z.B. Fitting darstellt?

Im Gegensatz zu den Tableau-Verfahren bei Fitting werden die Tableau-Regeln nicht nur dann ausgeführt, wenn eine bestimmte Formel im Zweig des Tableaus steht, sondern auch nur dann, wenn eine bestimmte Formel (noch) nicht im Tableau steht. Das führt zu einer Vermeidung von Endlosschleifen, weil die Expansion strikt ist.

27) Was mag der Grund dafür sein, dass Baader & Nutt bei dem Tableau-Ansatz von der Negations-Normalform ausgehen? Welche Konsequenz hat das für die Prüfung des Abschlusses eines Tableau-Zweiges?

Wir haben keine Negationsregel zum Expandieren von Tableaus. Dadurch, dass die Negationen immer innen zu finden sind ($\neg A$), ist garantiert, dass wir nur atomare Abschlüsse bilden. Zur Erstellung der Negationsnormalform wendet man die üblichen dualen Regeln (de Morgan, etc.) an, die wir aus der Prädikatenlogik kennen (\geq ist dual zu \leq). Ein atomarer Abschluss kann auch aufwendiger sein, wenn innerhalb einer Formel Widersprüche aufgeführt sind und diese Widersprüche erst nach einer Expansion erkannt werden.

28) Wie wird die Korrektheit der Tableau-Expansion begründet? Was lässt sich zur Korrektheit der einzelnen Expansionsregeln sagen?

Wir prüfen bei Tableau-Beweisen mit Beschreibungslogiken nur Konsistenz. Diese wird erhalten, weil die Expansionsregeln das gewährleisten.

29) Wie wird die Termination des Tableau-Verfahrens begründet? Welche interne Struktur weisen die in der Expansion von $C_0(x_0)$ konstruierten ABoxen jeweils auf?

Wir fangen mit $\{C(a)\}$ an. Alle Verzweigungen und Expansionen ergeben Formeln, die weniger komplex sind. Es gibt nur endlich viele Verzweigungen und Expansionen. Das ist daran zu erkennen, dass mit \exists und \geq nur endlich viele Relationen (und bei \exists zusätzlich Instanzen) erstellt werden. Die \forall -Regel verwendet nur die $R(x,y)$, die von diesen beiden Expansionsregeln bereits gebildet wurden, und die Instanzbildung $C(y)$ der \forall -Regel benutzt dieselben Konstanten

wie bei der Bedingung der Regel.

Wir stellen fest, dass $\top \sqsubseteq \text{CTBox}$. Alle neuen ABox-Einträge werden in dem Konzept CTBox festgehalten. Auch $D1 \sqsubseteq D2 \rightarrow \top \sqsubseteq D1 \sqcup \neg D2$ wird aufsummiert, bis die erstgenannte Inklusion stimmt. Genaueres zu generellen Inklusionsaxiomen wurde in der Sitzung vom 15.11.2006 geklärt und kann im folgenden Protokoll nachgelesen werden.